

# SSS-EC: Secure Storage Services and Erasure Code Implementation in Cloud Computing

<sup>1</sup>M. Abirami, <sup>2</sup>N.M. Nandhitha, <sup>3</sup>S.Emalda Roslin

<sup>1</sup>Final year software engineering student, Sathyabama University

<sup>2</sup>Head/ Academics, Dept. of ECE, Sathyabama University

<sup>3</sup>Head/ Academics, Dept. of E&C, Sathyabama University

**Abstract**—Cloud computing is sharing of resources on a larger scale which is cost effective and location independent. The level of security is low in cloud server for data safety. If at all security exists, the third party auditor should be allowed to access the entire data packets. In this paper a new cloud computing technique SSS-EC is proposed in which a Cloud server splits into different chunks and then encrypted. And the encrypted cloud server is kept in a replica cloud server as a backup. This encrypted data are converted into bytes and added parity bit process by the data owner in order to restrict TPA by accessing the original data.

**Keywords**—security, encryption, erasure code implementation, authentication.

## I INTRODUCTION

Cloud Computing means "Internet based Computing." The Internet is commonly visualized as clouds; hence the term "cloud computing" for computation done through the Internet. It is a technology that uses the internet and central remote servers to maintain data and application [1],[2]. Computing is a general term for anything that involves delivering hosted services over the Internet. In Figure1 the services are broadly divided into three categories: Infrastructure-as-a-Service (IaaS), Platform as-a-Service (PaaS) and Software-as-a-Service (SaaS).

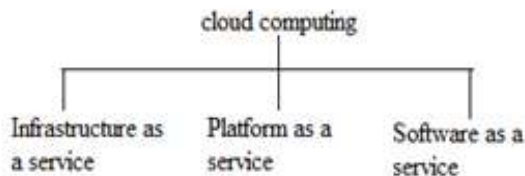


Figure1. Cloud computing delivery models

Infrastructure as a service sometimes referred as HaaS or hardware as a service. IaaS includes storage, hardware, servers and networking components. IaaS Provides user computing resources and storage comprised with many servers as an on-demand and "pay per use" service: Data Center, Bandwidth, Private Line Access, Servers and Server Room, Firewall PaaS model provides a platform for creating applications. PaaS solutions are essentially development platforms for which the development tool itself is hosted in the Cloud and accessed through a browser .With PaaS, developers can build Web applications without installing any tools on their computers.

PaaS bundles all stack components (hardware, infrastructure, storage) together with database, security, workflow, user interface, and other tools that allow users to create and host powerful business applications, web sites, and mobile apps. In the SaaS model, cloud providers install and operate application software in the cloud and cloud users access the software from clients. SaaS can be defined through five key ideas: services are fully managed and hosted, have regular recurring payments, allow for anytime and anywhere access, have multiple tenants on servers, don't require installation of specialized software. In Figure2 Cloud computing deployment models are classified into four categories: public cloud, private cloud, hybrid cloud, community cloud. Public cloud computing environment are open for use to anyone who wants to sign up and use them. A private cloud is basically an organization that needs more control over their data than they can get by using a vendor hosted service. A hybrid cloud combines both public and private cloud models.

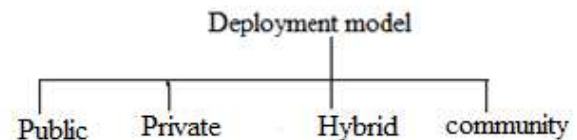


Figure2. Cloud computing deployment model

Data security protecting a database from destructive forces and the unwanted actions of unauthorized users. Security and privacy issues in cloud computing are identity management, availability, physical and personnel ability, application security. Cloud computing security is an evolving sub-domain of computer security, network security, and, more broadly, information security.[3][4]

In this paper, the various cloud computing models in the literature is surveyed. A comparison table is also made on evaluating the existing protocols. The paper is organized as follows: A comparative analysis is made in the chapter 2. chapter 3 gives a various data security in cloud computing in the literature is discussed. In chapter 4, the existing work in the implementation of data security is discussed. In the Chapter 5, data integrity and implementation of erasure technique. Conclusion and future work is given in chapter 6.

II COMPARISON TABLE

S No	Algorithm Used	Metrics used	Limitation	Advantages
5	Provable Data Possession Model (PDP)	1)dynamic provable data possession(DPDP) 2)version control system(CVS)	PDP scheme apply only to the static (or append-only) files	Support provable updates to stored data.
6	Erasur code technique	Peer to peer architecture method	Imposes large storage and network overhead.	High probability of retrieving files under severe peer failure conditions
7	Classic Merkle Hash Tree construction	Verify the integrity of the dynamic data stored in the cloud	Remote data storage mainly focus on static data files	1) Efficient data dynamics 2) Support efficient handling of multiple auditing tasks
8	Principle of secret sharing and erasure Coding	Data Integrity verification Scheme exploiting the technique of algebraic signatures	1) Individual sensors are not Reliable 2) Byzantine failures and Can be compromised due to lack of tamper-proof hardware.	1) Data con-fidentiality and dependability 2) Efficient dynamic data integrity checking
9	Multiple-Replica Provable Data Possession (MR-PDP)	Single-replica PDP scheme to store t separate	Generation of further replicas, on demand is little expensive	1)Replica maintenance 2)File checking
10	Cryptographic protocol	Public key distribution and digital signature	1)Protocol is its vulnerability to both centralized Loss of security and centralized loss of function. 2)Theft of the central keys	Security and reliability of centralized key distribution
11	Java implementation of the encoding algorithm of the new variant	1)New variant on the Juels-Kaliski protocol 2) Design of POR protocols	Parity blocks do not fit into main memory. 2) Problems of designing efficient POR protocols that support file updates,	1) Supports efficient protocols. 2) Tolerates higher error rates 3) Secure in a stronger adversarial setting
12	Privacy preserving auditing and extraction of digital contents	Initialization, audit, and extraction	Protocols do not protect against denial of service attacks.	Removes the burden of verification from the customer
13	Principles of RAID(Redundant Arrays of Inexpensive Disks)	HAIL (High-Availability and Integrity Layer)	HAIL only provide assurance for static files	1) Efficiency, security, and modeling improvement.
14	Incremental algorithm	Incremental scheme for hashing and signing	Problem is to design an incremental signature scheme secure against substitution attack	Efficiency for cryptographic transformation.
15	Internet-based backup technique	1)Periodic random challenges to ensure partners continue to hold data	Inevitable free rider and disrupter attacks to which a cooperative system is vulnerable.	Feasible and very inexpensive
16	BFT(Byzantine Fault Tolerance)	1) Proactive Recovery 2) Use of symmetric cryptography to authenticate Messages.	BFT does not rely on synchrony to provide safety.	1) BFT provides safety and liveness regardless of the number of Byzantine-faulty clients. 2)Detection of denial-of-service attacks

### III RELATED WORK

In [5] author used the provable data possession (PDP) model, the client preprocesses the data and then sends it to an untrusted server for storage, while keeping a small amount of meta-data. A definitional framework and efficient constructions for dynamic provable data possession (DPDP), which extends the PDP model to support provable updates on the stored data is provided. A DPDP scheme is proposed to an outsourced file systems and version control systems. The integrity of data stored at untrusted servers is prevented.

In [6] author proposed the peer to peer architecture model, is widely used for distributed systems deployment, which is increasingly becoming used as a model for disk backup systems. Erasure coding parameters with respect to peer availability and file size based on this analysis is provided. Erasure coding is an alternative failure handling scheme for replication. And it ensures data redundancy.

In [7] author explained the task of allowing a third party auditor (TPA), on behalf of the cloud client, to verify the integrity of the dynamic data stored in the cloud. TPA can perform multiple auditing tasks simultaneously. The problem of ensuring the integrity of data storage in Cloud Computing is prevented. To achieve efficient data dynamics, the existing proof of storage models by manipulating the classic Merkle Hash Tree construction for block tag authentication.

In [8] author used the principle of secret sharing and erasure coding. Based on the principle of secret sharing and erasure coding, a hybrid share generation and distribution scheme to achieve reliable and fault-tolerant initial data storage by providing redundancy for original data components. The scheme enables individual sensors to verify in one protocol execution all the pertaining data shares simultaneously in the absence of the original data. The security and performance analysis against various attacks is reduced.

In [9] author explained, multiple-replica provable data possession (MR-PDP). MR-PDP extends previous work on data possession proofs for a single copy of a file in client/server storage system. And verify through a challenge-response protocol that each unique replica can be produced at the time of the challenge and the storage system uses  $t$  times the storage required to store a single replica. The problem of creating multiple unique replicas of a file in a distributed storage system is gained. Replica Maintenance, file checking and can generate further replicas on demand is reduced.

In [10] author introduced, public key distribution and digital signature are compared with each other and with the conventional alternative. The author prevents the security and reliability of centralized key distribution. The author explained about the number of cryptographic protocols.

In [11] author introduced a new variant on the Juels-Kaliski protocol and describes a prototype implementation. A theoretical framework is designed for PORs. Finally concludes

that the Java implementation of the encoding algorithm of the new variant, in which files are processed and encoded incrementally.

In [12] author explained about privacy-preventing. They never reveal the data contents to the auditor. Protocols that allow a third party auditor to periodically verify the data stored by a service and assist in returning the data intact to the customer. The solution removes the burden of verification from the customer. The two main contributions are first the motivation, assumptions, and setting for third-party privacy preserving auditing and extraction, and second various privacy-preserving protocols for auditing and extraction of data stored with a service provider.

In [13] author introduced HAIL (High-Availability and Integrity Layer), a distributed cryptographic system that allows a set of servers to prove to a client that a stored file is intact and retrievable. A strong, formal adversarial model for HAIL is proposed. The two basic approaches to client verification of file availability and integrity. The HAIL is a remote-file integrity checking protocol that offers efficiency, security, and modeling improvements over straightforward multi-server application of POR protocols.

In [14] author explained incremental scheme for hashing and signing. Some basic definition enabling treatment of the new notion. A new kind of efficiency for cryptographic transformation. The incremental algorithm for cryptographic transformation. The problem is to design an incremental signature scheme secure against substitution attack.

In[15] author proposed a new Internet-based backup technique that appears to be one to two orders of magnitude cheaper than existing Internet backup services. A novel peer-to-peer backup technique that allows computers connected to the Internet to back up their data cooperatively. The problem against the inevitable free rider and disrupter attacks to which a cooperative system is vulnerable.

In [16] author described a new replication algorithm BFT, which can be used to build highly available systems that tolerate Byzantine faults. A proactive recovery mechanism that allows the replicated system to tolerate any number of faults over the lifetime of the system. The most important optimization is the use of symmetric cryptography to authenticate messages. Detection of denial-of-service attacks aimed at increasing the window and detects when the state of a replica is corrupted by an attacker is provided.

### III EXISTING WORK

In the existing system data owner sends a data to the cloud server using bidirectional way. To access the data from the Cloud server, the users have to be registered with the cloud server. So that the user have to register their details like username, password and a set of random numbers. This information will store in the database for the future

authentication. Once the Data Owner registered in cloud server, the space will be allotted to the Data Owner. There is no high security provided in the Cloud server for data safety. If at all security exists, the third party auditor should access the entire data packets for verification. Third party auditors are used by clients and providers to determine the security of the cloud implementation. The main drawback is there is no backup process and data safety.

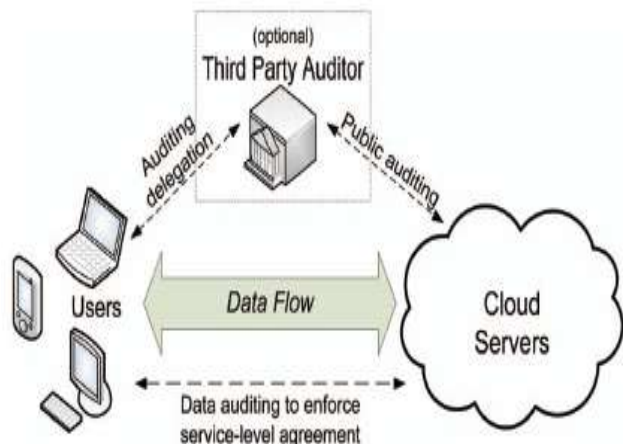


Figure 3. Cloud storage architecture of the existing system  
IV PROPOSED SYSTEM

In the proposed system, Main Cloud server spilt the file into batches and then encrypted. The corresponding encrypted batches are kept in different Cloud servers and their keys are distributed in different key server. So that we can increase the security of the cloud network. If the user wants retrieve the data, they've to provide the entire key that is stored in the appropriate key servers.

These encrypted batches are kept in replica servers as a backup. Suppose the data in the data server was lost, then the Main Cloud server will contact the Replica Cloud server and get the data from the Replica Cloud Server. This encrypted data are converted into bytes and added parity bit process by the data owner in order to restrict TPA by accessing the original data. Also we're applying the Erasure Code by using the XOR operation, while XORing the block data, the data will be converted in binary data.

Once added the parity added bits, then the data will be given to the Trusted Parity auditor. The Trusted Parity Auditor will generate the signature using change and response method

The Cloud server generates the token number from the parity added encrypted data and compared with the signature provided to the TPA to verify the Data Integrity.

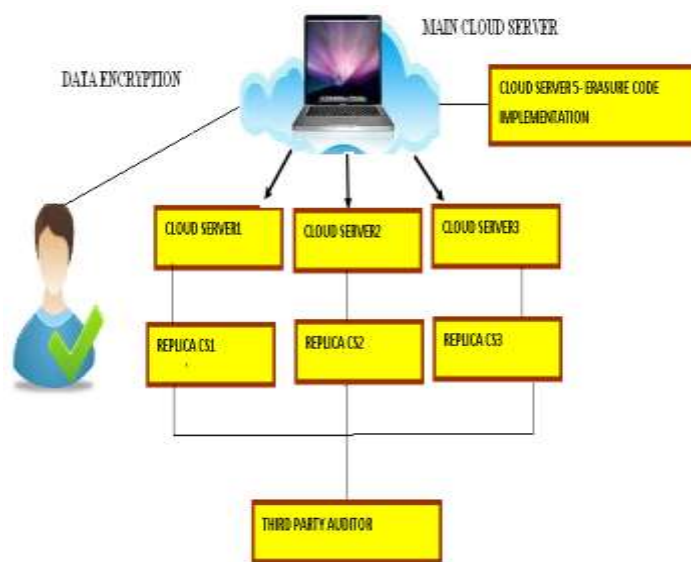


Figure 4. Cloud storage architecture of the proposed system  
Modules:

The proposed work incorporates seven different modules. They are

- Data owner
- Main cloud server
- Data splitting and encryption
- Key server
- Replica server
- Parity bit addition
- Trusted party auditor and erasure code

*a) Dataowner*

Data Owner is the Person who is going to upload the data in the Cloud Server. To upload the data into the Cloud server, the Data Owner have be registered in the Cloud Server. Once the Data Owner registered in cloud server, the space will be allotted to the Data Owner.

*b) Main cloud server*

Cloud Server is the area where the user going to request the data and also the data owner will upload their data. Once the user send the request regarding the data they want, the request will first send to the Cloud Server and the Cloud Server will forward your request to the data owner. The data Owner will send the data the data the user via Cloud Server. The Cloud Server will also maintain the Data owner and Users information in their Database for future purpose.



```

C:\Windows\system32\cmd.exe

C:\Users\nanaskar\Desktop\Data Integrity>javac *.java
Note: * uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: MainWindow.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\Users\nanaskar\Desktop\Data Integrity>java MainServer
Server Started....!
Constructor Started Here...
Server Started ...
    
```

Figure 5. Main server

```

C:\Windows\system32\cmd.exe

C:\Users\nanaskar\Desktop\Data Integrity>del *.bak
Could Not Find C:\Users\nanaskar\Desktop\Data Integrity\*.bak

C:\Users\nanaskar\Desktop\Data Integrity>javac *.java
Note: * uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: MainWindow.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\Users\nanaskar\Desktop\Data Integrity>java CloudServer2
CloudServer2 Server Started.... at port No: 333
    
```

Figure 7. Cloud server2

*c) Data splitting and encryption:*

The data was uploaded into the cloud server; the Cloud server will split the data into many parts and store all the data in the separate data servers. In techniques wasn't used in proposed system so that there might be a chance of hacking the entire data. Avoid the hacking process, we are splitting the data and store those data in corresponding data server. We're also encrypting the data segments before storing into the data server.

In Figure6 cloud server1 is started at port no: 222. In Figure7 cloud server2 is represented in the port no: 333. In Figure8 cloud server3 given in the port no: 444. In Figure9 cloud server4 is started at port no: 555

```

C:\Windows\system32\cmd.exe

C:\Users\nanaskar\Desktop\Data Integrity>del *.bak
Could Not Find C:\Users\nanaskar\Desktop\Data Integrity\*.bak

C:\Users\nanaskar\Desktop\Data Integrity>javac *.java
Note: * uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: MainWindow.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\Users\nanaskar\Desktop\Data Integrity>java CloudServer3
CloudServer3 Server Started.... at port No: 444
    
```

Figure 8. Cloud server3

```

C:\Windows\system32\cmd.exe

C:\Users\nanaskar\Desktop\Data Integrity>del *.bak
Could Not Find C:\Users\nanaskar\Desktop\Data Integrity\*.bak

C:\Users\nanaskar\Desktop\Data Integrity>javac *.java
Note: * uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: MainWindow.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\Users\nanaskar\Desktop\Data Integrity>java CloudServer1
CloudServer1 Server Started.... at port No: 222
    
```

Figure 6. Cloud server1

```

C:\Windows\system32\cmd.exe

C:\Users\nanaskar\Desktop\Data Integrity>del *.bak
Could Not Find C:\Users\nanaskar\Desktop\Data Integrity\*.bak

C:\Users\nanaskar\Desktop\Data Integrity>javac *.java
Note: * uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: MainWindow.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\Users\nanaskar\Desktop\Data Integrity>java CloudServer4
CloudServer4 Server Started.... at port No: 555
    
```

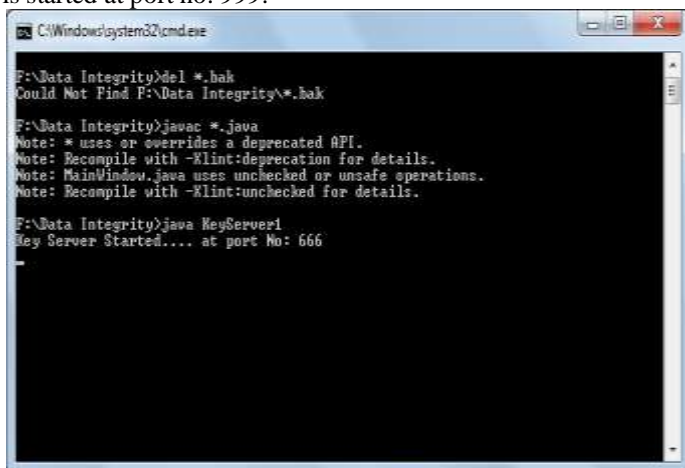
Figure 9. Cloud server4

*d) Key server*

The encryption keys are stored in appropriate key servers. So that we can increase the security of the cloud network. If the

user wants retrieve the data, they've to provide the entire key that is stored in the appropriate key servers.

In Figure10 key server1 is generated at port no: 666. In Figure11 key server2 is represented at port no: 777. In Figure12 key server3 is shows the port no: 888. In Figure13 key server4 is started at port no: 999.

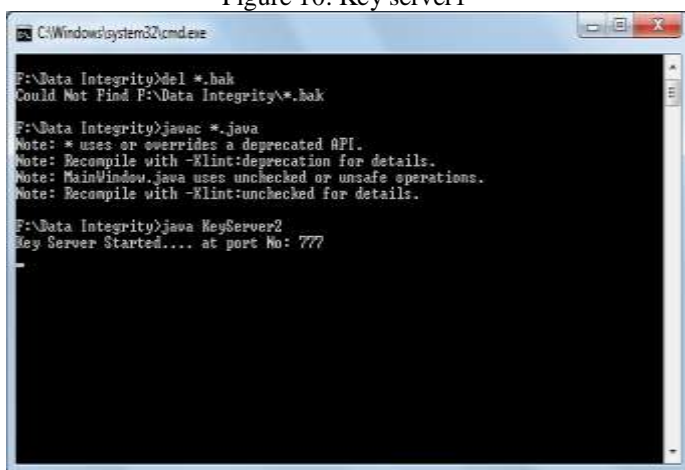


```
C:\Windows\system32\cmd.exe
F:\Data Integrity>del *.bak
Could Not Find F:\Data Integrity\*.bak

F:\Data Integrity>javac *.java
Note: * uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: MainWindow.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

F:\Data Integrity>java KeyServer1
Key Server Started.... at port No: 666
```

Figure 10. Key server1

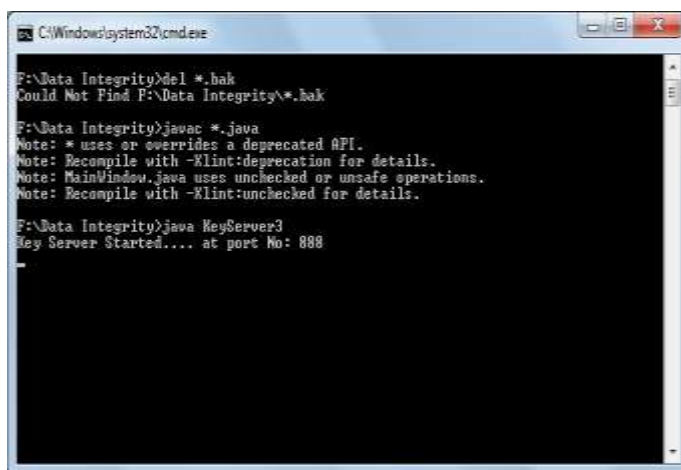


```
C:\Windows\system32\cmd.exe
F:\Data Integrity>del *.bak
Could Not Find F:\Data Integrity\*.bak

F:\Data Integrity>javac *.java
Note: * uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: MainWindow.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

F:\Data Integrity>java KeyServer2
Key Server Started.... at port No: 777
```

Figure 11. Key server2

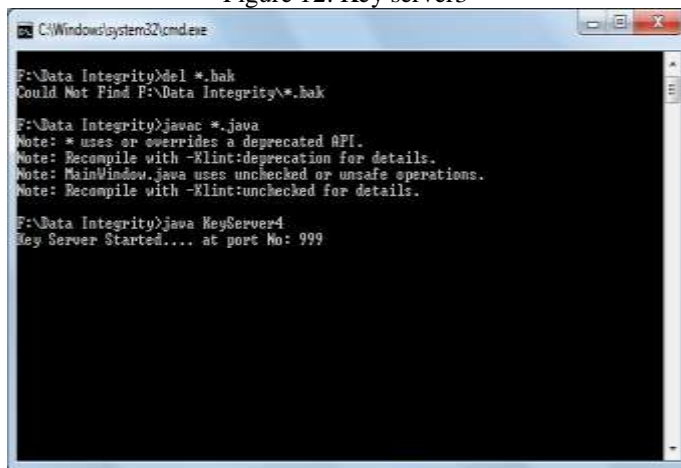


```
C:\Windows\system32\cmd.exe
F:\Data Integrity>del *.bak
Could Not Find F:\Data Integrity\*.bak

F:\Data Integrity>javac *.java
Note: * uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: MainWindow.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

F:\Data Integrity>java KeyServer3
Key Server Started.... at port No: 888
```

Figure 12. Key server3



```
C:\Windows\system32\cmd.exe
F:\Data Integrity>del *.bak
Could Not Find F:\Data Integrity\*.bak

F:\Data Integrity>javac *.java
Note: * uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: MainWindow.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

F:\Data Integrity>java KeyServer4
Key Server Started.... at port No: 999
```

Figure 13. Key server4

e) Replica Server:

Replica Cloud server is maintained. Suppose the data in the data server was lost, then the Main Cloud server will contact the Replica Cloud server and get the data from the Replica Cloud Server. By using this concept, we can get the data if any data loss occurs.

In Figure14 replica cloud server1 is generated at port no: 1111. In Figure15 replica cloud server2 is started at port no: 2222. In Figure16 replica cloud server3 is given in the port no: 3333. In Figure17 replica cloud server4 is represented at port no: 4444

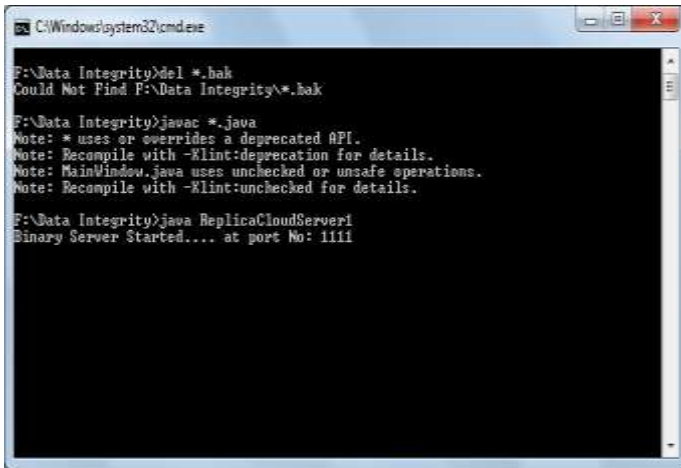


Figure 14. Replica cloud server1

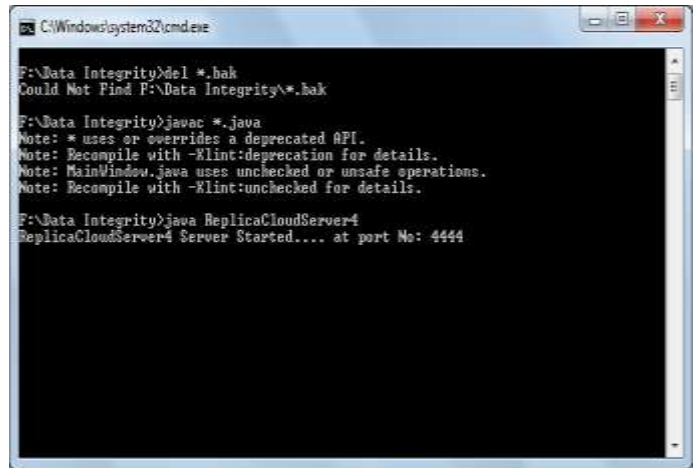


Figure 17. Replica cloud server4

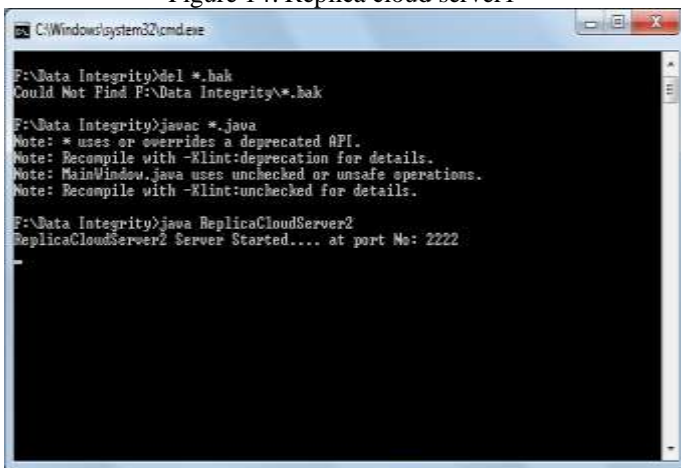


Figure 15. Replica cloud server2

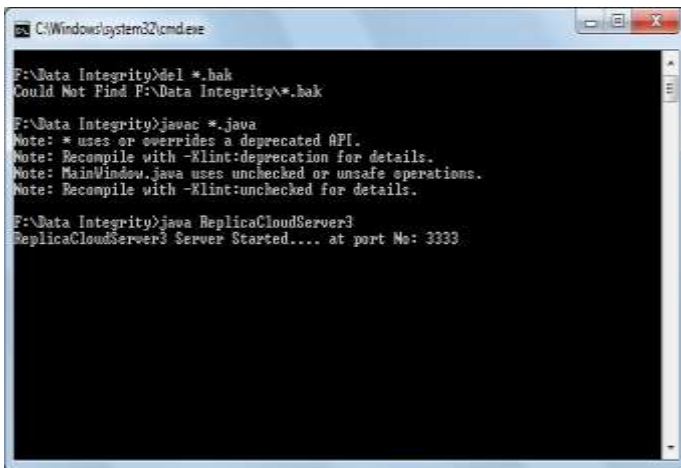


Figure 16. Replica cloud server3

To access the data from the Cloud server, the users have to be registered with the cloud server. So that the user have to register their details like username, password and a set of random numbers. This is information will stored in the database for the future authentication.



Figure 18. User registration

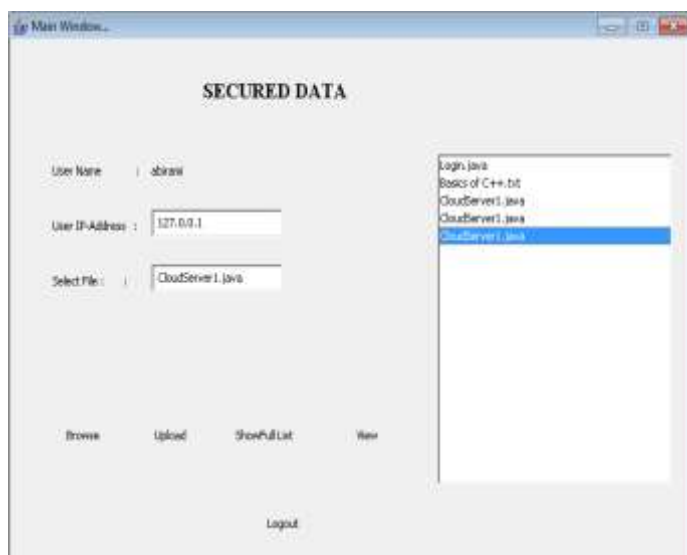


Figure 19. File upload

In Figure20 path is created for the selected file and values are updated

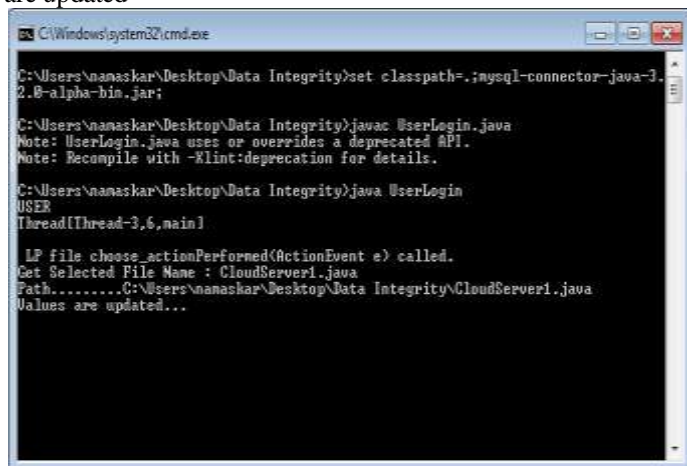


Figure 20. Creating path for the file and file uploaded

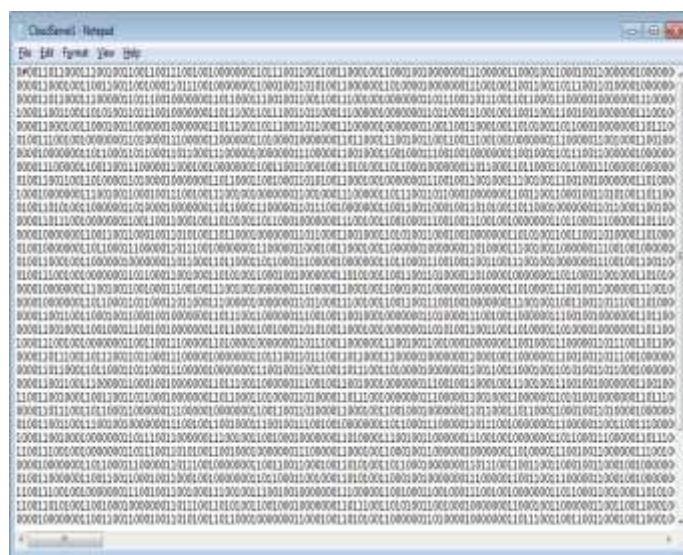


Figure 21. Byte and parity bit

### VI CONCLUSION

In this paper, we investigated the problem of data security in cloud data storage, which is essentially a distributed storage system. To achieve the assurances of cloud data integrity and availability and enforce the quality of dependable cloud storage service for users, we propose an effective and flexible distributed scheme with explicit dynamic data support, including block update, delete, and append. We rely on erasure-correcting code in the file distribution preparation to provide redundancy parity vectors and guarantee the data dependability. By utilizing the homomorphic token with distributed verification of erasure coded data, our scheme achieves the integration of storage correctness insurance and data error localization, i.e., whenever data corruption has been detected during the storage correctness verification across the distributed servers, we can almost guarantee the simultaneous identification of the misbehaving server.

### VII REFERENCES

- [1] <http://dotnetslackers.com/articles/sql/Introduction-to-Cloud-Computing.aspx>
- [2] [http://www.wikinest.com/concept/Cloud\\_Computing](http://www.wikinest.com/concept/Cloud_Computing)
- [3] [en.wikipedia.org/wiki/Cloud\\_computing](http://en.wikipedia.org/wiki/Cloud_computing)
- [4] [http://en.wikipedia.org/wiki/Data\\_security](http://en.wikipedia.org/wiki/Data_security)
- [5] C. Erway, A. Kupcu, C.Papamanthou, and R. Tamassia, "Dynamic Provable Data Possession," Proc. 16th ACM Conf. Computer and Comm. Security (CCS '09), pp. 213-222, 2009.
- [6] George Nychis, Argyro Andreou, Deepti Chheda, and Alexander Giamas, "Analysis of Erasure Coding in a Peer to Peer Backup System," "IEEE Trans. on information networking, 2008



[7] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing," IEEE Trans. Parallel and Distributed Systems, vol. 22, no. 5, pp. 847-859, 2011.

[8] Q. Wang, K. Ren, W. Lou, and Y. Zhang, "Dependable and Secure Sensor Data Storage with Dynamic Integrity Assurance," Proc. IEEE INFOCOM, Apr. 2009.

[9] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP: Multiple-Replica Provable Data Possession," Proc. IEEE 28th Int'l Conf. Distributed Computing Systems (ICDCS '08), pp. 411-420, 2008.

[10] R.C. Merkle, "Protocols for Public Key Cryptosystems," Proc. IEEE Symp. Security and Privacy, 1980.

[11] K.D. Bowers, A. Juels, and A. Oprea, "Proofs of Retrievability: Theory and Implementation," Proc. ACM Workshop Cloud Computing Security (CCSW '09), pp. 43-54, 2009.

[12] M.A. Shah, R. Swaminathan, and M. Baker, "Privacy-Preserving Audit and Extraction of Digital Contents,"

CryptologyePrint Archive, Report 2008/186, <http://eprint.iacr.org>, 2008.

[13] K.D. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage," Proc. ACM Conf. Computer and Comm. Security (CCS '09), pp. 187-198, 2009.

[14] M. Bellare, O. Goldreich, and S. Goldwasser, "Incremental Cryptography: The Case of Hashing and Signing," Proc. 14th Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '94), pp. 216-233, 1994.

[15] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard, "A Cooperative Internet Backup Scheme," Proc. USENIX Ann. Technical Conf. (General Track), pp. 29-41, 2003.

[16] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance and Proactive Recovery," ACM Trans. Computer Systems, vol. 20, no. 4, pp. 398-461, 2002.